ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA171

CONTROL SYSTEM SOFTWARE DEVELOPMENT ENVIRONMENT IN ELI BEAMLINES

P. Bastl†, O. Janda, A. Kruchenko, P. Pivonka, B. Plötzeneder, S. Saldulkar, J. Trdlicka
ELI Beamlines/Institute of Physics of the ASCR, Prague, Czech Republic

Abstract

The ELI Beamlines facility is a Petawatt laser facility in the final construction and commissioning phase in Prague, Czech Republic. End 2017, a first experiment will be performed. In the end, four lasers will be used to control beamlines in six experimental halls. The central control system connects and controls more than 40 complex subsystems (lasers, beam transport, beamlines, experiments, facility systems, safety systems), with high demands on network, synchronisation, data acquisition, and data processing. It relies on a network based on more than 15.000 fibres, which is used for standard technology control (PowerLink over fibre and standard Ethernet), timing (WhiteRabbit) and dedicated high-throughput data acquisition. Technology control is implemented on standard industrial platforms (B&R) in combination with uTCA for more demanding applications. Approach to software development is very important in such a facility. Component based generic approach described is efficient for both, software development and also software deployment.

INTRODUCTION

ELI Beamlines [1] is an emerging high-energy, high-repetition rate laser facility located in Prague, Czech Republic. Four laser beamlines (ranging from the inhouse developed L1 with <20fs pulses exceeding 100mJ at 1kHz based on DPSS technology to the 10PW-L4, developed by National Energetics) will supply six experimental halls which provide various secondary sources to users. Facility commissioning, and installation work of lasers and experiments is progressing, and first user experiments are expected in 2018.

The central control system connects, supervises and controls all technical installations used for the operation of this facility, which are more than 40 complex subsystems (lasers, beam transport, beamlines, experiments, plant systems (HVAC, vacuum), safety systems) with high demands on network, synchronisation, data acquisition, processing, and storage.

This paper describes the control system software development environment in ELI Beamlines.

APPROACH

There are three factors that make the development of the ELIs' control system challenging:

First, ELI has been designed to be multifunctional, and to provide a highly diverse selection of lasers and secondary sources to researchers. In practise this means that we have to integrate a **multitude of very diverse subsystems** developed by internal and external suppliers; leading to an initially very inhomogeneous technical landscape, and complex system interfaces.

At the same time, ELI is **building ground breaking technology** and its demands on synchronisation and data acquisition are pushing the boundaries on what is possible with current technology. Demands are especially high on safe operation, synchronization, and data acquisition.

Third, in ELI **commissioning and operational phases overlap**. While one part of the facility is still under development, others are being installed, and again others will be already serving early users (whose experiments need to be supported technically, and for whom laser and beam transport operation, safety, timing and data acquisition services must be provided).

We are using following approaches in our software development environment and architecture:

- **Use already available frameworks** for software development. In our case we chose TANGO framework which is available for free and is widely used in many facilities around the world [2].
- **Use only few programming languages** in our case we are using one compiled language which is the C++ and one scripting language which gives opportunity for the users to create their own scripts for experiments, here we chose Python which naturally support object orinted development. Both languages are of course supported by TANGO framework. Also standard user tools like Matlab, LabVIEW and others are supported by providing bindings for software components.
- **Using standard development process** based on modelling language which is UML and SysML in our case. The software development process is based on Iterative Unified Process
- **Use model based development focusing components and well defined API.** Here we use Enterprise architect [3].
- **Use automatic software generators** where possible. Communication with instrumentations

† pavel.bastl@eli-beams.eu

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA171

based on SCPI and VX-11 interfaces are good examples.

- **Use XML configuration** for each component. This gives as the opportunity to keep the basic API simple and clean and also hide specific parametrization in XML file.
- **Use software plugins.** Software plugins bring another level of generalization into the software. Plugins allows flexibility when the component is in operation and brings also flexibility to the user

SOFTWARE DEVELOPMENT PROCESS

Our software development process is based on the Iterative Unified Process. The approach breaks software into single components that go through iterations of architecture, design, implementation and testing. On top of the process is Redmine [4] project management tool and build and version management tools (git, cmake). Standard software build structure with automated version numbers taken from git repository. Also modern software engineering techniques such as continuous integration are used. The overall view of the process is shown on Figure 1.

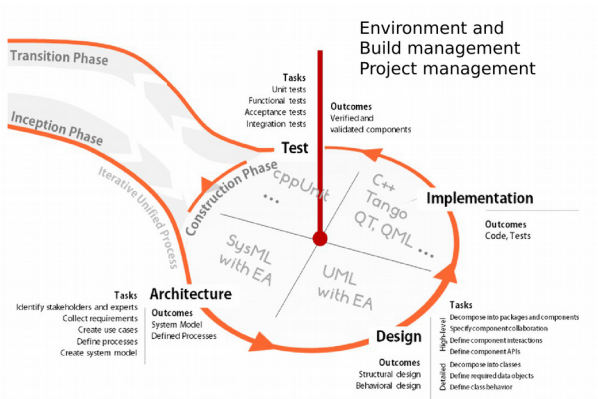

Figure 1: Software development process.

CONTROL SYSTEM SOFTWARE ARCHITECTURE AND DESIGN

The control system architecture and designs are created in Enterprise architect tool. The software is divided into software components. Each software component uses one or more defined abstract API and can use other APIs. All APIs are purely abstract classes and the component itself implements them.

Software Component

The generic structure of software component is shown on Figure 2. The structure depicts the following main approaches:

- Component implements only abstract APIs
- TANGO interface is developed based on the abstract API, therefore it is possible to write the TANGO server just once and reuse the API. This requires compilation
- Using Plugin allows to omit compilation process and can be used in run time if allowed

Every component can have also specific parameters. Consider for example cameras. There exists many camera types which can have various parameters. All the parameters can be described inside specific XML file format. Therefore the generic component structure can be configured by its own XML file. The XML configuration can play two basic roles:

- Allows configuration of specific hardware or component. The configuration may be saved in local file or in inventory database. The solution with database allows allows the user to edit the configuration and reload when in appropriate state.
- Allows to write generic GUI. In this case the configuration is serialized in the component and send to GUI where the XML is parsed and the GUI automatically generates form for configuration parameters.

The XML parametrization is provided by specific implementation of selected abstract API and therefore may change when another plugin is used even during operation of the component. Good example is GUI for camera.

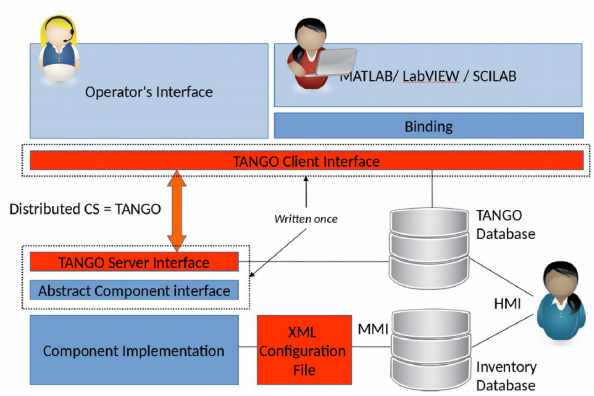

Figure 2: Generic structure of software component.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA171

GRAPHICAL USER INTERFACE

We have developed graphical user interface in ELI Beamlines which supports all the techniques used in the generic software component. These techniques include standard GUI interface and automatically generated form for specific parametrization.

Figure 3: Graphical user interface

The GUI is shown on Figure 3. The GUI is written in Qt/Qml and allows user to build his own panel based on components. The components can be arranged inside the main area. The area setup can be saved in XML file and open later by the user on another computer. In the picture is also shown TANGO browser where appropriate TANGO interface can be selected and connected to the GUI. XML configuration file contains also all information about selected TANGO devices, so if the appropriate TANGO database is available all of them can be automatically connected. Also Python based scripting interface is provided (not shown on the picture).

OUTLOOK

In the upcoming years, lasers and secondary sources will be gradually installed and put into operation, which will be certainly challenging for the control systems team.

Specifically in 2018, we are expecting to work on the interfaces to two lasers (L1 / L3) and to control a number of secondary sources in their early stage of operation (for example ELIMAIA , HHG, an Ellipsometer, TEREZA, PXS and the MAC chamber with mostly vision/mottion control and detectors. The L3 beam transport will also provide another challenge with a very high number of controllable devices. The software is developed in advance and adapted to real conditions of operation. During this time, both the implementation and API can be still changed. Later all the API shall matured and stable. We are also expecting that new API can be defined during deployment of the software. This case influences also design of the software but corresponds to the iterative software development approach.

We trust that matured API can bring a lot of advantages for software development at any facility and also for potential manufacturers of devices. If stable API is available then suppliers of technologies used in big physics facilities can relay on it and simplify integration of devices. It has to be noted that the integration is important during development of control system and also for potential user who can bring his own hardware. The configuration of components through XML files allows to use even specific parameters.

CONCLUSION

This paper gave an overview of software development environment in ELI Beamlines with description of some techniques like abstract API and plugins. These techniques are mainly bringing flexibility for software development and also for the users. Some of them also significantly simplify development and provide place for further development.

REFERENCES

[1] ELI Beamlines, https://www.eli-beams.eu
[2] TANGO Controls, http://www.tango-controls.org/
[3] Enterprise architect, https://www.sparxsystems.com.au
[4] Redmine, https://www.redmine.org/